Web Components
Hola Mundo

•••

La función de un buen software es hacer
que lo complejo aparente ser simple

Web Components

Eres mejor programador de lo que piensas.

Web Component

Web Component - Registro

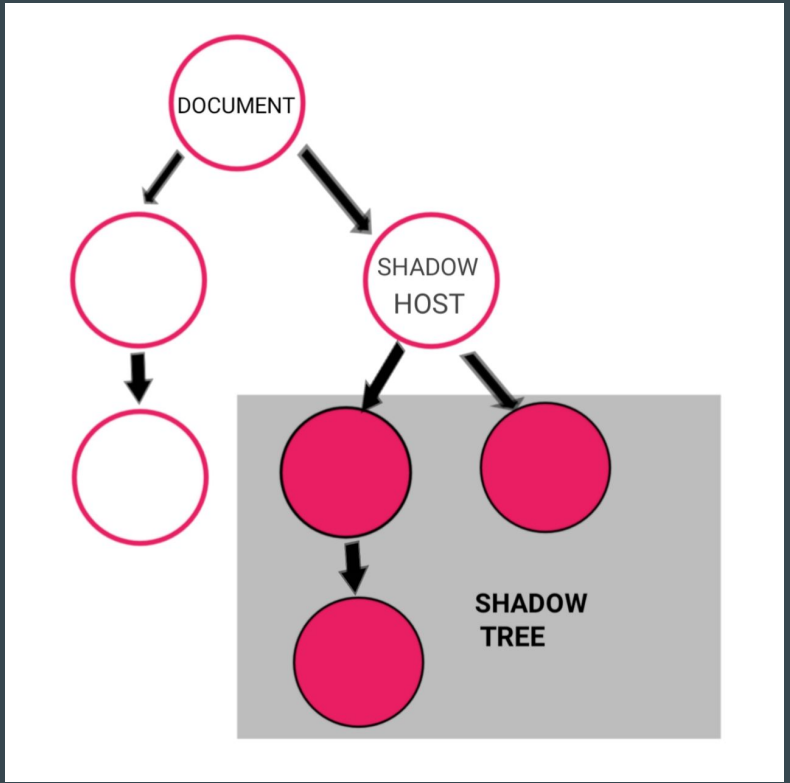
- Herencia clase HTMLElement
- Registro del componente en customElements
- Uso del tag registrado

```
<body>
    <like-button color="red"></like-button>

    <script>
        class LikeButton extends HTMLElement {···

        // Registrar el componente
        customElements.define("like-button", LikeButton);
    </script>
</body>
```

Web Component - Shadow DOM

- Shadow DOM

```
class LikeButton extends HTMLElement {
    constructor() {
        super();
        this.attachShadow({ mode: "open" });

        // Estado inicial
        this.count = 0;
        this.color = this.getAttribute("color") || "blue";
    }
```



Web Component - Atributos

- Las propiedades que entran al componente se leen con “getAttribute”

- Cuidado, no se pueden pasar funciones como atributo, hay que pasarlas en Javascript

- La función “observedAttributes” hace que el componente reaccione a cambios en el atributo.
- La función “attributeChangedCallback” se ejecuta cada vez que cambian los atributos

Web Component - Render

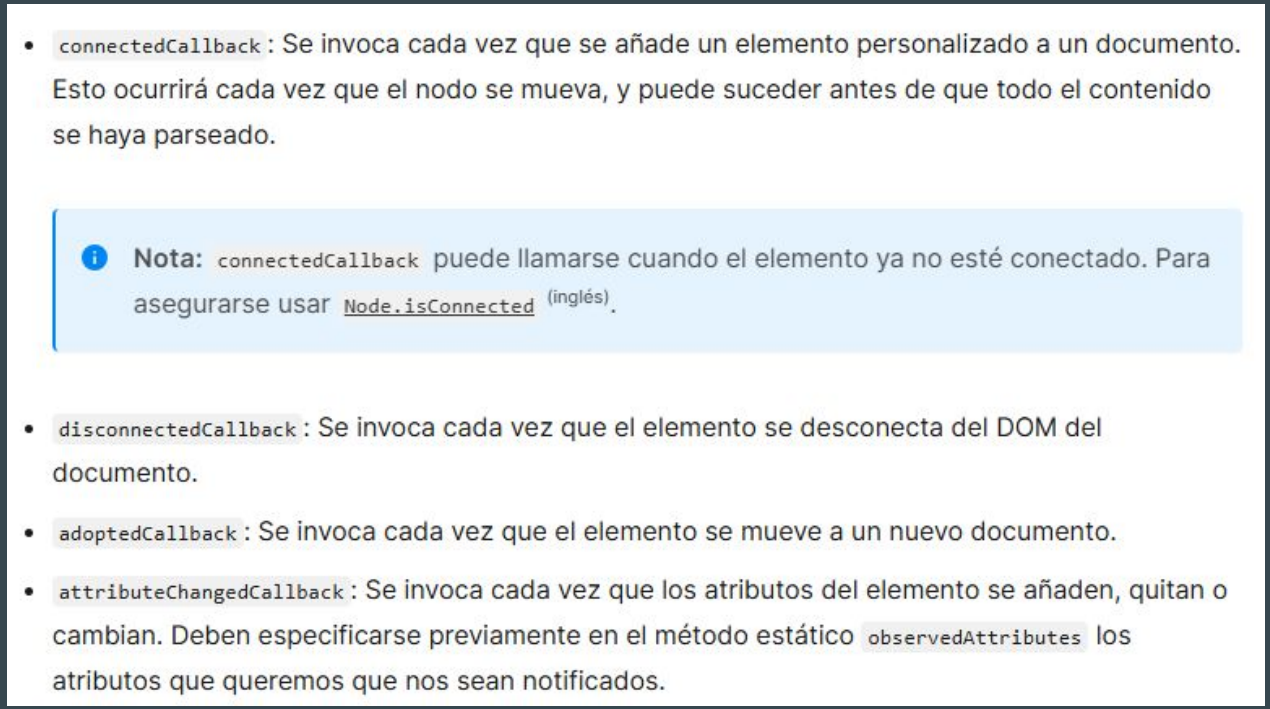
- La función render es la que contiene nuestro HTML + CSS

- Se suele poner en la func “attributeChangedCallback” para que se repinte cada vez que hay un cambio.

```
render() {
    // Crear el template del botón
    this.shadowRoot.innerHTML = /* html */ `
        <style>
            button {
                background-color: ${this.color};
                color: white;
                border: none;
                padding: 10px;
                border-radius: 5px;
                cursor: pointer;
                font-size: 16px;
            }
        </style>
        <button>❤️ 0</button>
    `;
}

attributeChangedCallback() {
    this.render();
}
```

Web Component - Ciclo de vida

- Ciclo de vida de un Web Component: Link



- `connectedCallback`: Se invoca cada vez que se añade un elemento personalizado a un documento. Esto ocurrirá cada vez que el nodo se mueva, y puede suceder antes de que todo el contenido se haya parseado.

  > **Nota:** `connectedCallback` puede llamarse cuando el elemento ya no esté conectado. Para asegurarse usar `Node.isConnected` (inglés).

- `disconnectedCallback`: Se invoca cada vez que el elemento se desconecta del DOM del documento.
- `adoptedCallback`: Se invoca cada vez que el elemento se mueve a un nuevo documento.
- `attributeChangedCallback`: Se invoca cada vez que los atributos del elemento se añaden, quitan o cambian. Deben especificarse previamente en el método estático `observedAttributes` los atributos que queremos que nos sean notificados.

Web Component - Javascript

- Añadiremos el Javascript de nuestro componente en la func “connectedCallback”
- El estado interno del componente lo tendremos como una variable privada de la clase como “count” en nuestro ejemplo que inicializamos en el constructor.

```
connectedCallback() {
    this.button = this.shadowRoot.querySelector("button");
    this.button.addEventListener("click", () => this.onClick());
}

increment() {
    this.count++;
    this.button.textContent = `❤️ ${this.count}`;
}
```